COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes
Number of questions—4
Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
- 1. A music Web site keeps track of downloaded music. For each download, the site uses a DownloadInfo object to store a song's title and the number of times it has been downloaded. A partial declaration for the DownloadInfo class is shown below.

The list of downloaded information is stored in a MusicDownloads object. A partial declaration for the MusicDownloads class is shown below.

```
public class MusicDownloads
  /** The list of downloaded information.
      Guaranteed not to be null and not to contain duplicate titles.
  private List<DownloadInfo> downloadList;
  /** Creates the list of downloaded information. */
  public MusicDownloads()
  { downloadList = new ArrayList<DownloadInfo>(); }
   /** Returns a reference to the DownloadInfo object with the requested title if it exists.
        @param title the requested title
        @return a reference to the DownloadInfo object with the
                  title that matches the parameter title if it exists in the list;
                  null otherwise.
       Postcondition:
         - no changes were made to downloadList.
    * /
  public DownloadInfo getDownloadInfo(String title)
  { /* to be implemented in part (a) */
   /** Updates downloadList with information from titles.
       Oparam titles a list of song titles
       Postcondition:
         - there are no duplicate titles in downloadList.
         - no entries were removed from downloadList.
         - all songs in titles are represented in downloadList.
         - for each existing entry in downloadList, the download count is increased by
              the number of times its title appeared in titles.
         - the order of the existing entries in downloadList is not changed.
         - the first time an object with a title from titles is added to downloadList, it
              is added to the end of the list.
         - new entries in downloadList appear in the same order
              in which they first appear in titles.
         - for each new entry in downloadList, the download count is equal to
              the number of times its title appeared in titles.
    * /
  public void updateDownloads(List<String> titles)
      /* to be implemented in part (b) */
  // There may be instance variables, constructors, and methods that are not shown.
```

Part (a) begins on page 4.

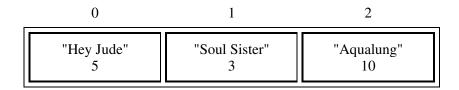
}

{

© 2013 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(a) Write the MusicDownloads method getDownloadInfo, which returns a reference to a DownloadInfo object if an object with a title that matches the parameter title exists in the downloadList. If no song in downloadList has a title that matches the parameter title, the method returns null.

For example, suppose variable webMusicA refers to an instance of MusicDownloads and that the table below represents the contents of downloadList. The list contains three DownloadInfo objects. The object at position 0 has a title of "Hey Jude" and a download count of 5. The object at position 1 has a title of "Soul Sister" and a download count of 3. The object at position 2 has a title of "Aqualung" and a download count of 10.



The call webMusicA.getDownloadInfo("Aqualung") returns a reference to the object in position 2 of the list.

The call webMusicA.getDownloadInfo("Happy Birthday") returns null because there are no DownloadInfo objects with that title in the list.

```
Class information repeated from the beginning of the question

public class DownloadInfo

public DownloadInfo(String title)
public String getTitle()
public void incrementTimesDownloaded()

public class MusicDownloads

private List<DownloadInfo> downloadList
public DownloadInfo getDownloadInfo(String title)
public void updateDownloads(List<String> titles)
```

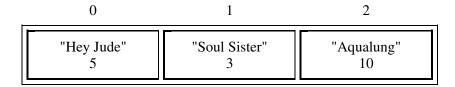
WRITE YOUR SOLUTION ON THE NEXT PAGE.

Complete method getDownloadInfo below.

Part (b) begins on page 6.

(b) Write the MusicDownloads method updateDownloads, which takes a list of song titles as a parameter. For each title in the list, the method updates downloadList, either by incrementing the download count if a DownloadInfo object with the same title exists, or by adding a new DownloadInfo object with that title and a download count of 1 to the end of the list. When a new DownloadInfo object is added to the end of the list, the order of the already existing entries in downloadList remains unchanged.

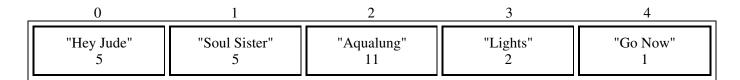
For example, suppose variable webMusicB refers to an instance of MusicDownloads and that the table below represents the contents of the instance variable downloadList.



Assume that the variable List<String> songTitles has been defined and contains the following entries.

{"Lights", "Aqualung", "Soul Sister", "Go Now", "Lights", "Soul Sister"}

The call webMusicB.updateDownloads (songTitles) results in the following downloadList with incremented download counts for the objects with titles of "Soul Sister" and "Aqualung". It also has a new DownloadInfo object with a title of "Lights" and a download count of 2, and another DownloadInfo object with a title of "Go Now" and a download count of 1. The order of the already existing entries remains unchanged.



Class information repeated from the beginning of the question

public class DownloadInfo

public DownloadInfo(String title)
public String getTitle()
public void incrementTimesDownloaded()

public class MusicDownloads

private List<DownloadInfo> downloadList
public DownloadInfo getDownloadInfo(String title)
public void updateDownloads(List<String> titles)

In writing your solution, you must use the getDownloadInfo method. Assume that getDownloadInfo works as specified, regardless of what you wrote for part (a).

© 2013 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Complete method updateDownloads below.

```
/** Updates downloadList with information from titles.
     Oparam titles a list of song titles
     Postcondition:
      - there are no duplicate titles in downloadList.
      - no entries were removed from downloadList.
      - all songs in titles are represented in downloadList.
      - for each existing entry in downloadList, the download count is increased by
            the number of times its title appeared in titles.
      - the order of the existing entries in downloadList is not changed.
      - the first time an object with a title from titles is added to downloadList, it
           is added to the end of the list.
      - new entries in downloadList appear in the same order
            in which they first appear in titles.
      - for each new entry in downloadList, the download count is equal to
            the number of times its title appeared in titles.
public void updateDownloads(List<String> titles )
```

2. A multiplayer game called Token Pass has the following rules.

Each player begins with a random number of tokens (at least 1, but no more than 10) that are placed on a linear game board. There is one position on the game board for each player. After the game board has been filled, a player is randomly chosen to begin the game. Each position on the board is numbered, starting with 0.

The following rules apply for a player's turn.

- The tokens are collected and removed from the game board at that player's position.
- The collected tokens are distributed one at a time, to each player, beginning with the next player in order of increasing position.
- If there are still tokens to distribute after the player at the highest position gets a token, the next token will be distributed to the player at position 0.
- The distribution of tokens continues until there are no more tokens to distribute.

The Token Pass game board is represented by an array of integers. The indexes of the array represent the player positions on the game board, and the corresponding values in the array represent the number of tokens that each player has. The following example illustrates one player's turn.

Example

The following represents a game with 4 players. The player at position 2 was chosen to go first.

			ţ	
Player	0	1	2	3
Player Tokens	3	2	6	10

The tokens at position 2 are collected and distributed as follows.

1st token - to position 3 (The highest position is reached, so the next token goes to position 0.)

2nd token - to position 0

3rd token - to position 1

4th token - to position 2

5th token - to position 3 (The highest position is reached, so the next token goes to position 0.)

6th token - to position 0

After player 2's turn, the values in the array will be as follows.

			ţ	
Player Tokens	0	1	2	3
Tokens	5	3	1	12

The Token Pass game is represented by the TokenPass class.

```
public class TokenPass
  private int[] board;
  private int currentPlayer;
   /** Creates the board array to be of size playerCount and fills it with
        random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
        random integer value in the range between 0 and playerCount-1, inclusive.
        @param playerCount the number of players
  public TokenPass(int playerCount)
  \{ /* \text{ to be implemented in part (a) } */ \}
   /** Distributes the tokens from the current player's position one at a time to each player in
        the game. Distribution begins with the next position and continues until all the tokens
        have been distributed. If there are still tokens to distribute when the player at the
        highest position is reached, the next token will be distributed to the player at position 0.
        Precondition: the current player has at least one token.
        Postcondition: the current player has not changed.
    * /
  public void distributeCurrentPlayerTokens()
      /* to be implemented in part (b) */ }
   // There may be instance variables, constructors, and methods that are not shown.
```

(a) Write the constructor for the TokenPass class. The parameter playerCount represents the number of players in the game. The constructor should create the board array to contain playerCount elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable currentPlayer to a random number between 0 and playerCount-1, inclusive.

Complete the TokenPass constructor below.

(b) Write the distributeCurrentPlayerTokens method.

The tokens are collected and removed from the game board at the current player's position. These tokens are distributed, one at a time, to each player, beginning with the next higher position, until there are no more tokens to distribute.

Class information repeated from the beginning of the question

public class TokenPass
private int[] board
private int currentPlayer
public TokenPass(int playerCount)
public void distributeCurrentPlayerTokens()

Complete method distributeCurrentPlayerTokens below.

- /** Distributes the tokens from the current player's position one at a time to each player in
- * the game. Distribution begins with the next position and continues until all the tokens
- * have been distributed. If there are still tokens to distribute when the player at the
- * highest position is reached, the next token will be distributed to the player at position 0.
- * **Precondition**: the current player has at least one token.
- * **Postcondition**: the current player has not changed.

* /

public void distributeCurrentPlayerTokens()